

IN THE CLAIMS

- 1 1. [previously amended] A process comprising:
2 validating a ~~formal language specification~~Formal Language
3 Specification written in a formal language which has predetermined rules of
4 syntax and semantics, said ~~formal language specification~~Formal Language
5 Specification defining a computer program to be automatically written;
6 automatically translating each element of a ~~formal language~~
7 ~~specification~~Formal Language Specification defining an object model, a
8 functional model, a dynamic model and a presentation model, which taken
9 together define the requirements of the program to be automatically written,
10 into a full and complete computer program which needs no additional third
11 party source code or source code from existing components or code libraries
12 to be compiled with it to make said computer program complete and which
13 implements the requirements of said ~~formal language specification~~Formal
14 Language Specification, said ~~formal language specification~~Formal Language
15 Specification defining at least an identification function for every class, and at
16 least a valuation for every variable attribute said translating step comprising the
17 following steps:
18 using a computer, automatically write computer code that will
19 request user name and password, receive any responses and
20 authenticate the user;
21 using a computer, automatically write computer code that has the
22 capability to control a computer to provide a mechanism to will
23 determine a this user's privilege level from log in information supplied
24 by said user which identifies said user and query said ~~formal language~~
25 ~~specification~~Formal Language Specification and determine all object
26 attributes said user has privilege to see and all services said user has
27 privileges to can invoke;

28 using a computer, automatically write computer code which has
29 the capability to query queries said ~~formal language specification~~ Formal
30 Language Specification for all services of all classes that any authorized
31 user has privileges to ~~may~~ invoke and identify ~~identifies~~ an object server
32 which will implement each said service;

33 using a computer, automatically write code that has the capability
34 to ~~will~~ retrieve service arguments for all services;

35 using a computer, automatically write code that is capable of
36 controlling ~~controls~~ a computer to provide a display means by which an
37 entity has a mechanism to ~~and entity can~~ invoke a service, and which
38 has a mechanism to receive ~~receives~~ input to invoke a particular service
39 and respond ~~responds~~ by sending a message to the appropriate object
40 server to invoke the service, said message including the ~~necessary~~
41 arguments required for the service to execute;

42 using a computer, automatically write code that has the capability
43 to control a computer to implement ~~implements~~ an object server for
44 every service, each of which first checks to verify that state transitions are
45 valid ~~and make sense~~ for the current state of objects of which the object
46 server will be altering the state;

47 using a computer, automatically write code that has the capability
48 to control a computer to implement ~~for~~ every object server that verifies
49 preconditions are satisfied before making state transitions of any
50 objects the states of which are acted upon by the object server;

51 using a computer, automatically write code that has the capability
52 to control a computer to make all valuation calculations required by said
53 ~~formal language specification~~ Formal Language Specification of each
54 object server;

55 using a computer, automatically write code that has the capability

56 to control a computer to verify that integrity constraints specified in said
 57 ~~formal language specification~~Formal Language Specification on the
 58 values of attributes of objects have been satisfied after execution of a
 59 service and respond by reversing any changes in state which caused
 60 said integrity constraint to be violated ~~take action~~ if said integrity
 61 constraints are not satisfied; and
 62 using a computer, automatically write code that has the capability
 63 to control a computer to implement ~~for every object server~~ such that said
 64 object server tests ~~to test~~ trigger relationships specified in said ~~formal~~
 65 ~~language specification~~Formal Language Specification after execution of
 66 a service and invoke a predetermined service associated with a trigger
 67 event ~~carry out appropriate action~~ if said a trigger event has occurred.

1 2. [currently amended] An apparatus for automatically translating a
 2 ~~formal language specification~~Formal Language Specification defining an
 3 object model, a functional model, a dynamic model and a presentation model,
 4 which taken together define the requirements of a computer program to be
 5 automatically written, said ~~formal language specification~~Formal Language
 6 Specification being written in a formal language which has predefined rules of
 7 grammar, said translating acting to convert said ~~formal language~~
 8 ~~specification~~Formal Language Specification into a computer program that
 9 implements the requirements of said ~~formal language specification~~Formal
 10 Language Specification, said ~~formal language specification~~Formal Language
 11 Specification defining at least an identification function for every class and at
 12 least a valuation for every variable attribute, said apparatus comprising:
 13 a computer programmed with an operating system and one or more
 14 other programs to cooperate with said operating system to control said
 15 computer to perform the following functions:

16 A) using said predetermined rules of grammar to validate said
17 ~~formal language specification~~Formal Language Specification to ensure
18 that said ~~formal language specification~~Formal Language Specification
19 is complete and correct;

20 B) automatically write computer code that will request user name
21 and password, receive any responses and authenticate the user;

22 C) automatically write computer code that will determine a user's
23 privilege level for a user identified by user name and password entered
24 in response to the query caused by the code written in step B, and query
25 said formal specification and determine all object attributes said user
26 has privilege to see and query and all services said user has privileges
27 to can-invoke;

28 D) automatically write computer code which queries said formal
29 specification for all services of all classes that any authorized user has
30 privileges to may-invoke and identifies an object server which will
31 implement said service;

32 E) automatically write code that will retrieve service arguments for
33 all services;

34 F) automatically write code that displays one or more user
35 interface tools which provide a mechanism ~~can be used~~ to invoke a
36 service, and which provides a mechanism to receive ~~receives~~ input to
37 invoke a particular service and which responds by sending a message
38 to the appropriate object server to invoke said service, said message
39 including the ~~necessary~~ arguments required for said service to execute;

40 G) automatically write code that implements an object server for
41 every service, each of which first checks to verify that state transitions are
42 valid ~~and make sense~~ for the current state of objects the object service
43 will be altering the state of;

H) automatically write code for every object server that verifies preconditions are satisfied before making state transitions of any objects the states of which are acted upon by said object server;

I) automatically write code to make all valuation calculations required by said specification of each object server;

J) automatically write code to verify that integrity constraints specified in said Formal Language Specification ~~formal specification~~ on the values of attributes of objects have been satisfied after execution of a service and reversing any changes in state which caused said integrity constraints to be not satisfied ~~take action~~ if said integrity constraints are not satisfied; and

K) automatically write code for every object server to test trigger relationships specified in said Formal Language Specification ~~formal specification~~ after execution of a service and invoke a predetermined service associated with a trigger event ~~carry out appropriate action~~ if a trigger event has occurred.

3. [currently amended] A physical computer-readable storage media ~~medium~~ containing instructions for controlling a computer to automatically translate a ~~formal language specification~~ Formal Language Specification defining an object model, a functional model, a dynamic model and a presentation model, which taken together define the requirements of a computer program to be automatically written, said ~~formal language specification~~ Formal Language Specification defining at least an identification function for every class and at least a valuation for every variable attribute said ~~formal language specification~~ Formal Language Specification written in a formal language having predefined rules of grammar, by:

validating a ~~formal language specification~~ Formal Language

12 Specification written in a formal language which has predetermined rules of
13 syntax and semantics, said validating accomplished using said predetermined
14 rules of syntax and semantics to ensure said ~~formal language~~
15 ~~specification~~ Formal Language Specification is complete and correct;
16 automatically writing computer code that will request user name and
17 password, receive any responses and authenticate the user;
18 automatically writing computer code that will determine a user's
19 privilege level and query said Formal Language Specification ~~formal~~
20 ~~specification~~ and determine all object attributes said user has privilege to see
21 and all services said user has privileges to ~~can~~ invoke;
22 automatically writing computer code which queries said Formal
23 Language Specification ~~formal specification~~ for all services of all classes that
24 any authorized user has privileges to ~~may~~ invoke and identifies an object server
25 which will implement said service;
26 automatically writing computer code that will retrieve service arguments
27 for all services;
28 automatically write code that displays menus options, icons or creates
29 any other means or mechanism through ~~by~~ which a user or another process
30 has the capability to ~~can~~ invoke a service, and which provides a mechanism
31 through which ~~receives~~ input to invoke a particular service is provided and
32 mechanisms to provide values for arguments is provided and a mechanism is
33 provided to construct and send ~~responds by sending~~ a message to the
34 appropriate object server to invoke the service, said message including the
35 ~~necessary~~ arguments required for the service to execute;
36 automatically writing code that implements an object server for every
37 service, each of which first checks to verify that state transitions are valid ~~and~~
38 ~~make sense~~ for the current state of objects the object service will be altering
39 the state of;

4 0 automatically write code for every object server that verifies preconditions
 4 1 are satisfied before making state transitions of any objects the states of which
 4 2 are acted upon by the object server;

4 3 automatically write code to make all valuation calculations required by
 4 4 said Formal Language Specification ~~formal specification~~ of each object server;

4 5 automatically write code to verify that integrity constraints specified in
 4 6 said Formal Language Specification ~~formal specification~~ on the values of
 4 7 attributes of objects have been satisfied after execution of a service and
 4 8 reverse any state changes which have been made which cause said integrity
 4 9 constraints to be not satisfied ~~take action~~ if said integrity constraints are not
 5 0 satisfied; and

5 1 automatically write code for every object server to test trigger
 5 2 relationships specified in said Formal Language Specification ~~formal~~
 5 3 ~~specification~~ after execution of a service and invoke a predetermined service
 5 4 associated with a trigger event ~~carry out appropriate action~~ if a trigger event
 5 5 has occurred.

1 4. [currently amended] An apparatus for automatically translating a
 2 Formal Language Specification written in any formal language defining a full
 3 and complete Conceptual Model of a desired computer program to be
 4 automatically generated into a full and complete source code which
 5 implements said desired computer program, comprising:

6 a computer programmed with an operating system and one or more
 7 other programs to cooperate with said operating system to control said
 8 computer to perform the following functions:

9 reading all said primitives in said Formal Language Specification

1 0 in any order;

1 1 in any order, using a computer and said Formal Language

12 Specification to automatically generate computer code which has the
13 capability to control a computer to carry out generating one or more
14 methods comprised of computer code which can control a computer to
15 perform the following functions in an order determined by an execution
16 model:

17 determining if said Formal Language Specification
18 requires user authentication, and, if so, automatically writing
19 computer code that will request user name and password,
20 receive any responses and authenticate the user;

21 determining if said Formal Language Specification
22 requires determining a user privilege level, and, if so,
23 automatically writing computer code that will determine a user's
24 privilege level and query said Formal Language Specification and
25 determine all object attributes said user has privilege to see and
26 determine all services this user has privileges to ~~can~~ invoke;

27 determining if said Formal Language Specification defines
28 services, and, if so, automatically writing computer code which
29 queries said Formal Language Specification to determine all
30 services that the authenticated user has privileges to ~~may~~ invoke
31 and which are defined in said Formal Language Specification for
32 all classes of objects said authenticated user will be able to view
33 and automatically writing an object server for each said service
34 which will implement said service upon receipt of a service
35 invocation message, each of said object servers containing code
36 which will perform the following functions in the following order
37 upon receipt of a service invocation message:

38 verify that one or more proposed state transitions
39 are valid ~~can be validly made~~ for the current state of any

40 object(s) of which said object server will be altering the
 41 state before actually altering the state of said object(s);
 42 verify that any preconditions of a service
 43 implemented by an object server are satisfied before said
 44 object server will act upon said one or more objects to
 45 make state transitions thereof in carrying out said service,
 46 ignoring said service invocation message if either
 47 said one or more state transitions cannot be validly made
 48 for the current state of any objects upon which said object
 49 server will be acting or any said precondition is not
 50 satisfied;
 51 if all said proposed transitions are valid ~~can be~~
 52 ~~validly made~~ on said one or more objects upon which said
 53 object server will act and if all said preconditions are
 54 satisfied, make all valuation calculations of said object
 55 server required by said Formal Language Specification;
 56 verify that service execution by said object server did
 57 not result in violation of one or more integrity constraints
 58 specified in said Formal Language Specification on the
 59 values of attributes of objects affected by execution of said
 60 service implemented by said object server, and take
 61 corrective action if one or more of said integrity constraints
 62 are not satisfied; and
 63 after a valid change of state of an object acted upon
 64 by said object server occurs, test trigger relationships or
 65 condition-action rules specified in said Formal Language
 66 Specification and, if any trigger event is satisfied, triggering
 67 a service specified in said condition-action rule or trigger

relationship.

5. [Cancelled]

6. [Cancelled]

7. [Cancelled]

8. [Cancelled]

10. [Currently amended] A process for automatically translating a Formal Language Specification written in any formal language defining a full and complete Conceptual Model of a desired computer program to be automatically generated into a full and complete source code which implements said desired computer program, comprising:

- reading all said primitives in said Formal Language Specification in any order;
- in any order, using a computer and said Formal Language Specification to automatically generate computer code that has the capability to control a computer to implement generating one or more methods which perform the following functions in an order determined by an execution model:
 - determining if said Formal Language Specification requires user authentication, and, if so, automatically writing computer code that will request user name and password, receive any responses and authenticate the user;
 - determining if said Formal Language Specification requires determining a user privilege level, and, if so, automatically writing computer code that will determine a user's privilege level and query said Formal Language Specification and determine all object attributes said user has privilege to see and determine all services this user has privileges to ~~can~~ invoke;
 - determining if said Formal Language Specification defines

24 services, and, if so, automatically writing computer code which
25 queries said Formal Language Specification to determine all
26 services that the authenticated user has privileges to ~~may~~ invoke
27 and which are defined in said Formal Language Specification for
28 all classes of objects said authenticated user will be able to view
29 and automatically writing an object server for each said service
30 which will implement said service upon receipt of a service
31 invocation message, each of said object server containing code
32 which will perform the following functions in the following order
33 upon receipt of a service invocation message:

34 verify that one or more proposed state transitions
35 are valid ~~can be validly made~~ for the current state of any
36 object(s) of which said object server will be altering the
37 state before actually altering the state of said object(s);

38 verify that any preconditions of a service
39 implemented by an object server are satisfied before said
40 object server will act upon said one or more objects to
41 make state transitions thereof in carrying out said service,

42 ignoring said service invocation message if either
43 said one or more state transitions cannot be validly made
44 for the current state of any objects upon which said object
45 server will be acting or any said precondition is not
46 satisfied;

47 if all said proposed transitions are valid ~~can be~~
48 ~~validly made~~ on said one or more objects upon which said
49 object server will act and if all said preconditions are
50 satisfied, make all valuation calculations of said object
51 server required by said Formal Language Specification;

52 verify that service execution by said object server did
53 not result in violation of one or more integrity constraints
54 specified in said Formal Language Specification on the
55 values of attributes of objects affected by execution of said
56 service implemented by said object server, and take
57 corrective action if one or more of said integrity constraints
58 are not satisfied; and

59 after a valid change of state of an object acted upon
60 by said object server occurs, test trigger relationships or
61 condition-action rules specified in said Formal Language
62 Specification and, if any trigger event is satisfied, triggering
63 a service specified in said condition-action rule or trigger
64 relationship.

1 11. [Cancelled]

1 12. [Cancelled]

1 13. [previously added] A process for converting a Formal Language
2 Specification encoding a Conceptual Model which defines desired system
3 logic and a desired user interface of a desired computer program into source
4 code which encodes said desired computer program, comprising:

5 validating said Formal Language Specification to ensure it is complete
6 and correct;

7 retrieving predetermined information from said Formal Language
8 Specification encoding a Conceptual Model which defines one or more
9 classes of objects in an object model, a dynamic model which specifies the
10 behavior of each object in response to services, triggers and global
11 transactions as represented by a state transition diagram for every class and
12 an object interaction diagram for every trigger and for every global transaction,
13 and a functional model which defines the semantics of any change of each

object's state as a consequence of an event occurrence by specifying for each class one or more mathematical or logical formulas which define how one or more variable attributes of said class will have their values changed when one or more specified events of said class occurs meaning one or more services of said class is executed;

using predetermined information retrieved from said Formal Language Specification to automatically generate source code which implements a presentation tier of said desired computer program;

using predetermined information retrieved from said Formal Language Specification to automatically generate source code which implements a persistence tier, database of data structure of said desired computer program;

using said predetermined information retrieved from said Formal Language Specification to automatically generate source code which implements a middle tier of said desired program which communicates with said presentation tier in the manner defined below, said middle tier written by automatically generating at least the following component instances for each class defined in said Formal Language Specification:

a server component instance including a method to implement each service present in a signature of said class and one or more methods to receive requests from said presentation tier that relate to execution of services of said class;

a query component instance including a method for implementing queries to extract information from said persistence tier relating to objects within said class and a method to receive and process requests from said presentation tier to query said persistence tier;

an executive component instance including one or more methods to receive and process a request from said server

42 component or another executive component to execute a service
43 in said class and carry out the following functions:

44 verify the existence and validity for a requested
45 server component;

46 create a copy of a requested server component
47 instance in memory and access said persistence tier
48 using said query component to retrieve values of constant
49 and variable attributes of said server component;

50 validate state transitions for said requested service
51 and a present state for a server component instance;

52 verify the satisfaction of preconditions specified in
53 said Formal Language Specification of said requested
54 service;

55 changing the state of said server component
56 instance to a new state by modifying a value of a variable
57 attribute of said server component instance by performing
58 all valuations specified in said functional model affected by
59 said requested service;

60 validating said new state by verifying said new state
61 does not violate static or dynamic restrictions specified in
62 said Formal Language Specification;

63 check trigger conditions established in said Formal
64 Language Specification to determine if said new state
65 causes any trigger to occur, and, if so, which actions
66 should be carried out;

67 communicate with said persistence tier to access or
68 store all attributes of said server component instance.

1 14. [currently amended] A process for validating a ~~formal language~~
2 ~~specification~~ Formal Language Specification, comprising the steps:

3 A) checking said ~~formal language specification~~ Formal Language
4 Specification to ensure that it is complete in that all required properties
5 of a Conceptual Model embodied in said ~~formal language~~
6 ~~specification~~ Formal Language Specification are defined and have a
7 valid value;

8 B) using rules of grammar of whatever formal language said
9 ~~formal language specification~~ Formal Language Specification is written
10 in, checking said ~~formal language specification~~ Formal Language
11 Specification to ensure it is correct in that it is syntactically and
12 semantically correct and not ambiguous.

1 15. [Previously added] The process of claim 14 further comprising the
2 step of presenting a request for said missing information via a mechanism of a
3 user interface if any information is missing or presenting a request via a user
4 interface mechanism to correct any syntactic or semantic error or clarify any
5 ambiguity discovered during said validation process.

1 16. [currently amended] The process of claim 14 further comprising the
2 step of doing a partial validation each time an element is added to said ~~formal~~
3 ~~language specification~~ Formal Language Specification to check for
4 completeness and correctness and mark the portion of said ~~formal language~~
5 ~~specification~~ Formal Language Specification just added as invalid if an error is
6 found so that a request to correct said error ~~is~~ can be presented later when a
7 full validation of said ~~formal language specification~~ Formal Language
8 Specification is requested.

1 17. [Previously added] The process of claim 14 wherein step A
 2 comprises checking to ensure that all elements in said Conceptual Model have
 3 a set of properties that exist and have a valid value and wherein step B
 4 comprises using a predetermined process and grammar for every type of
 5 formula in said Conceptual Model to ensure each is syntactically and
 6 semantically correct.

1 18. [Previously added] The process of claim 14 wherein step A
 2 comprises performing strict validation on some element properties and flexible
 3 validation of other element properties of said Conceptual Model, said strict
 4 validation of a property defined as requiring a full definition and valid value for a
 5 property, and flexible validation defined as allowing a property to be incomplete
 6 or have an invalid value during a process of inputting elements which define
 7 said Conceptual Model, and wherein the following table defines which
 8 elements and properties have strict or flexible validations:

Element	Property	Subproperty	Validation Type
class			
	name		strict
	ID function		flexible
	attributes (at least one)		flexible
	services (at least a Create service)		flexible

	static and dynamic integrity constraints	their formulas	strict
attribute			
	name		strict
	type (constant, variable, derived)		strict
	data-type (real, integer, etc.)		strict
	default value		strict
	size		strict
	request in creation service		strict
	null value allowed		strict
	evaluations (variable attributes)		flexible
	derivation formula (derived attributes)		flexible
Evaluation			

PATENT

	one variable attribute of a class		strict
	one service of the same class		strict
	condition		strict
	formula of evaluation		strict
derivation			
	formula		strict
	condition (optional)		strict
service			
	name		strict
	arguments		
		argument's name	strict
		data type	strict
		default value (optional)	strict
		null value	strict
		size (if proceeds)	strict

		formula of transaction	flexible
preconditions of an action			
	formula		strict
		agents affected by condition	strict
relationship: aggregation			
	related classes (component and composite)		strict
	relationship name		strict
	both directions: role names		strict
	cardinality		strict
	inclusive or referential		strict
	dynamic		strict
	clause "group by" (optional)		strict

PATENT

	insertion and deletion events (if proceed)		strict
relationship: inheritance			
	related classes (parent & child)		strict
	temporal (versus permanent)		strict
	specialization condition or events		strict
relationship: agent			
	agent class and service allowed to activate		strict
state transition diagram			
	all states of classes (three at least)		flexible

state in state transition diagram			
	name		strict
transition in state transition diagram			
	estate of origin		strict
	estate of destination		strict
	service of class		strict
		control condition (optional)	strict
trigger			
	condition		strict
	class or instance of destination		strict
	target (self, object, class)		strict
	activated service		strict

	service arguments' initialization (optional)		
		argument values	strict
global interactions			
	name		strict
	formula		strict
user exit functions			
	name		strict
	return data-type		strict
	arguments, (optional)		
	argument's name		strict
	argument's data-type		strict.

- 1 19. [Previously added] The process of claim 14 wherein step B
2 comprises validating formulas to ensure each formula complies with a precise
3 syntax defined for a formula of that type and is semantically correct, where there
4 are several types of formulas, and wherein a predetermined process for

5 validation and a set of rules of grammar exist for each type of formula and a
 6 validation process and a set of rules appropriate for the type of formula being
 7 validated is used for validation of each formula.

1 20. [currently amended] The process of claim 14 wherein step B
 2 comprises validating formulas to ensure each formula complies with a precise
 3 syntax defined for a formula of that type and is semantically correct, where there
 4 are several types of formulas defined by the table below:

default value calculation of	
	class attributes (constant and variable)
	service and transaction arguments
inheritance: specialization conditions	
static and dynamic integrity constraints	
derivations and valuations	
	effect formula (derived or variable attributes respectively)
	conditions (optional)
preconditions for actions	
control conditions for transitions in state transition diagram	
triggering conditions	

local and global transactions formulas	
---	--

1 and further comprising the steps of:

2 presenting dialog boxes via a user interface by which a user is
3 provided a mechanism to ~~may~~ enter said formulas during a process of
4 defining said Conceptual Model;
5 and wherein step B is performed by preventing a user from leaving a dialog box
6 being used to define a formula until said formula being defined is syntactically
7 and semantically correct.

1 21. [Previously added] The process of claim 14 wherein steps A and B
2 are performed by at least checking to ensure that every formula is syntactically
3 correct, every class has an identification function and has a creation event and
4 a destroy event, every triggering formula is semantically correct, every name of
5 an aggregation is unique in the scheme of said Conceptual Model, every
6 derived attribute has at least a derivation formula, every service has an agent or
7 server declared to execute it.

1 22. [currently amended] The process of claim 14 further comprising
2 performing steps A and B each time a user working on defining said
3 Conceptual Model makes a change which may invalidate one or more
4 formulas, but wherein predetermined formulas are allowed to be temporarily
5 incorrect so that the user is presented them for ~~can~~ review ~~them~~ at a later time
6 if they are still incorrect when a full validation process is performed after the
7 user is done defining the Conceptual Model.

1 23. [currently amended] The process of claim 14 further comprising the

2 step of using a computer to automatically translate said ~~formal language~~
3 ~~specification~~ Formal Language Specification into computer code after steps A
4 and B and been completed and all formulas are syntactically and semantically
5 complete and correct.

1 24. [Previously added] The process of claim 14 further comprising the
2 steps:
3 presenting user interface tools by which a user may define said
4 Conceptual Model and make changes thereto;
5 checking all affected formulas each time a change is made to
6 said Conceptual Model;
7 if the change affects a strictly validated property, then the change
8 is rejected if the property is not given a valid value, otherwise the change
9 is accepted;
10 if the change affects a property which is not strictly validated, then
11 the user is informed should any error arise, but allowed to do the
12 modification if said user ~~he or she~~ wishes;
13 if there are no affected formulas, modifying the Conceptual Model
14 as specified by the user.

1 25. [currently amended] The process of claim 14 wherein user interface
2 defining portions of said ~~formal language specification~~ Formal Language
3 Specification are validated by:
4 verifying that any patterns defined by said user are acceptable
5 user interface patterns with no essential information missing;
6 attributes used in filters specified as part of said user interface
7 are visible from a definition class;
8 attributes used in order criteria are visible from a definition class;

any formula in a filter is syntactically and semantically correct and
 uses only terms defined in said Conceptual Model;
 any action selection pattern uses as final actions object defined in
 said Conceptual Model;
 any set of dependency patterns are terminal and have confluence;
 and
 warnings are displayed to said user if any pattern is defined but not used or if
 an instance pattern is duplicated.

26. [currently amended] An apparatus comprising a computer
 programmed with an operating system and a validation program that
 cooperates with said operating system to control said computer to perform the
 following functions of a validation process:

A) check a ~~formal language specification~~ Formal Language
 Specification for completeness by checking to ensure said ~~formal
 language specification~~ Formal Language Specification has no missing
 information which is needed to detail the requirements of a desired
 computer program modelled by said ~~formal language
 specification~~ Formal Language Specification;

B) cooperate with said operating system to ensure said ~~formal
 language specification~~ Formal Language Specification is correct by
 checking each statement in said ~~formal language specification~~ Formal
 Language Specification according to rules of grammar of whatever
 formal language in which said ~~formal language specification~~ Formal
 Language Specification is written to ensure that each statement is
 syntactically and semantically correct and not ambiguous so as to
 ensure that all properties of elements in a Conceptual Model encoded in
 said ~~formal language specification~~ Formal Language Specification have

20 a value which is valid and to ensure that all formulas in said Conceptual
21 Model have correct syntax and meaning.

1 27. [Previously added] The apparatus of claim 26 wherein said validation
2 program controls said computer to perform the following additional function:
3 presenting a request for said missing information via a
4 mechanism of a user interface if any information is missing or
5 presenting a request via a user interface mechanism to correct any
6 syntactic or semantic error or clarify any ambiguity discovered during
7 said validation process.

1 28. [currently amended] The apparatus of claim 26 wherein said
2 validation program controls said computer to perform the following additional
3 function:
4 doing a partial validation each time an element is added to said
5 ~~formal language specification~~ Formal Language Specification to check
6 for completeness and correctness and mark the portion of said ~~formal~~
7 ~~language specification~~ Formal Language Specification just added as
8 invalid if an error is found so that a request to correct said error ~~is can be~~
9 presented later when a full validation of said ~~formal language~~
10 ~~specification~~ Formal Language Specification is requested if said error
11 still exists.

1 29. [Previously added] The apparatus of claim 26 wherein said validation
2 program controls said computer to perform the following additional function:
3 checking to ensure that all elements in said Conceptual Model
4 have a set of properties that exist and have a valid value;
5 performing step B by using a predetermined process and

6 grammar for every type of formula in said Conceptual Model to ensure
7 each is syntactically and semantically correct.

1 30. [Previously added] The apparatus of claim 26 wherein said
2 validation program controls said computer to perform step B by validating
3 formulas to ensure each formula complies with a precise syntax defined for a
4 formula of that type and is semantically correct, where there are several types
5 of formulas, and wherein a predetermined process for validation and a set of
6 rules of grammar exist for each type of formula and a validation process and a
7 set of rules appropriate for the type of formula being validated is used for
8 validation of each formula.

1 31. [Previously added] The apparatus of claim 26 wherein said
2 validation program controls said computer to perform step B by validating
3 formulas to ensure each formula complies with a precise syntax defined for a
4 formula of that type and is semantically correct, where there are several types
5 of formulas defined by the table below:

default value calculation of	
	class attributes (constant and variable)
	service and transaction arguments
inheritance: specialization conditions	
static and dynamic integrity constraints	
derivations and valuations	

	effect formula (derived or variable attributes respectively)
	conditions (optional)
preconditions for actions	
control conditions for transitions in state transition diagram	
triggering conditions	
local and global transactions formulas	

1 and wherein said validation program is further structured to control said
2 computer to perform the steps of:

3 presenting dialog boxes via a user interface by which a user may
4 enter said formulas during a process of defining said Conceptual Model;
5 and wherein step B is performed by preventing a user from leaving a dialog box
6 being used to define a formula until said formula being defined is syntactically
7 and semantically correct.

1 32. [Previously added] The apparatus of claim 26 wherein said validation
2 program controls said computer to perform the following steps:

3 presenting user interface tools by which a user may define said
4 Conceptual Model and make changes thereto;

5 checking all affected formulas each time a change is made to
6 said Conceptual Model;

7 if the change affects a strictly validated property, then the change
8 is rejected if the property is not given a valid value, otherwise the change
9 is accepted;

10 if the change affects a property which is not strictly validated, then
 11 the user is informed should any error arise, but allowed to do the
 12 modification if said user ~~he or she~~ wishes;

13 if there are no affected formulas, modifying the Conceptual Model
 14 as specified by the user.

1 33. [currently amended] ~~An~~ A physical computer readable storage media
 2 ~~medium~~ having stored thereon computer-readable instructions which when
 3 executed by a computer implement a validation program to control said
 4 computer to perform the following functions of a validation process:

5 A) check ~~a formal language specification~~ Formal Language
 6 Specification for completeness by checking to ensure said ~~formal~~
 7 ~~language specification~~ Formal Language Specification has no missing
 8 information which is needed to detail the requirements of a desired
 9 computer program modelled by said ~~formal language~~
 10 ~~specification~~ Formal Language Specification;

11 B) cooperate with said operating system to ensure said ~~formal~~
 12 ~~language specification~~ Formal Language Specification is correct by
 13 checking each statement in said ~~formal language specification~~ Formal
 14 Language Specification according to rules of grammar of whatever
 15 formal language in which said ~~formal language specification~~ Formal
 16 Language Specification is written to ensure that each statement is
 17 syntactically and semantically correct and not ambiguous so as to
 18 ensure that all properties of elements in a Conceptual Model encoded in
 19 said ~~formal language specification~~ Formal Language Specification have
 20 a value which is valid and to ensure that all formulas in said Conceptual
 21 Model have correct syntax and meaning.

1 34. [Previously added] The computer-readable medium of claim 33
2 wherein said validation program is further structured to control said computer
3 to perform the following additional function:

4 presenting a request for said missing information via a
5 mechanism of a user interface if any information is missing or
6 presenting a request via a user interface mechanism to correct any
7 syntactic or semantic error or clarify any ambiguity discovered during
8 said validation process.

1 35. [currently amended] The computer-readable medium of claim 33
2 wherein said validation program is further structured to control said computer
3 to perform the following additional function:

4 doing a partial validation each time an element is added to said
5 ~~formal language specification~~ Formal Language Specification to check
6 for completeness and correctness and mark the portion of said ~~formal~~
7 ~~language specification~~ Formal Language Specification just added as
8 invalid if an error is found so that a request to correct said error ~~is can be~~
9 presented later when a full validation of said ~~formal language~~
10 ~~specification~~ Formal Language Specification is requested if said error
11 still exists.

1 36. [Previously added] The computer-readable medium of claim 33
2 wherein said validation program is further structured to control said computer
3 to perform the following additional functions:

4 checking to ensure that all elements in said Conceptual Model
5 have a set of properties that exist and have a valid value;
6 performing step B by using a predetermined process and
7 grammar for every type of formula in said Conceptual Model to ensure

8 each is syntactically and semantically correct.

1 37. [Previously added] The computer-readable medium of claim 29
 2 wherein said validation program is structured to control said computer to
 3 perform step B by validating formulas to ensure each formula complies with a
 4 precise syntax defined for a formula of that type and is semantically correct,
 5 where there are several types of formulas, and wherein a predetermined
 6 process for validation and a set of rules of grammar exist for each type of
 7 formula and a validation process and a set of rules appropriate for the type of
 8 formula being validated is used for validation of each formula.

1 38. [Previously added] The computer-readable medium of claim 33
 2 wherein said validation program is structured to control said computer to
 3 perform step B by validating formulas to ensure each formula complies with a
 4 precise syntax defined for a formula of that type and is semantically correct,
 5 where there are several types of formulas defined by the table below:

default value calculation of	
	class attributes (constant and variable)
	service and transaction arguments
inheritance: specialization conditions	
static and dynamic integrity constraints	
derivations and valuations	

	effect formula (derived or variable attributes respectively)
	conditions (optional)
preconditions for actions	
control conditions for transitions in state transition diagram	
triggering conditions	
local and global transactions formulas	

1 and wherein said validation program is further structured to control said
2 computer to perform the steps of:

3 presenting dialog boxes via a user interface by which a user may
4 enter said formulas during a process of defining said Conceptual Model;
5 and wherein step B is performed by preventing a user from leaving a dialog box
6 being used to define a formula until said formula being defined is syntactically
7 and semantically correct.

1 39. [Previously added] The computer-readable medium of claim 33
2 wherein said validation program is structured to control said computer to
3 perform the following steps:

4 presenting user interface tools by which a user may define said
5 Conceptual Model and make changes thereto;
6 checking all affected formulas each time a change is made to
7 said Conceptual Model;
8 if the change affects a strictly validated property, then the change
9 is rejected if the property is not given a valid value, otherwise the change

is accepted;

if the change affects a property which is not strictly validated, then the user is informed should any error arise, but allowed to do the modification if said user ~~he or she~~ wishes;

if there are no affected formulas, modifying the Conceptual Model as specified by the user.

Please add the following new claims directed to a process to translate the validated formal language specification into computer code.

40. [New] A process to automatically translate a Formal Language Specification defining the functionality of a computer application modelled in a Conceptual Model, into a computer program called an application, said process comprising the steps of:

A) validating said Formal Language Specification to ensure said Formal Language Specification is complete in that there is no missing information in said Formal Language Specification and to ensure said Formal Language Specification is correct in that primitives of said conceptual model are syntactically and semantically consistent and not ambiguous;

B) translating said validated Formal Language Specification into computer readable code which has the capability to control a computer to provide a user interface access mechanism to allow users to log in by entering at least identification data and to use said identification data to authenticate and validate a user as an instance of a class of the validated fFormal Language Specification that act as agent in at least one agent relationship;

C) translating said validated Formal Language Specification into computer readable code which has the capability to control a computer to provide a view of the system defining the set of objects and attributes

the user can query and the set of services said user can execute, the content of said system view depending on the identity of said user accessing said application;
and

D) translating said validated Formal Language Specification into computer readable code which has the capability to control a computer to provide user interface interaction mechanisms to interact with and execute the functionality of the application in terms of performing queries on information managed by said application and executing services to modify the state of said information managed by said application, said services comprising events, local transactions and global transactions.

41. [New] The process of claim 40 wherein Step D comprises translating said Formal Language Specification into computer code which has the capability to control a computer so as to implement functionality of said queries as defined in said validated Formal Language Specification by means of filter formulas of filter patterns representing said queries, and so as to implement functionality of said services as defined in said validated Formal Language Specification in terms of:

- at least a valuation for each variable attribute of each class in said validated Formal Language Specification associated to at least an event of said class, which altogether define the functionality of every event;
- a transaction formula that defines a composition of services into a molecular execution unit, thereby defining functionality of every local transaction and global transaction;
- state transitions to control the valid lives for objects of each class in said validated Formal Language Specification, upon occurrence of an event or a local transaction;
- optional preconditions to the execution of services; and
- optional integrity constraints to prevent the execution of services from leaving the information managed by said application in an inconsistent or invalid state.

42. [New-Access To The System] The process of claim 40 wherein said step B creates computer readable code which has the capability to control a computer to provide user interface access

4 mechanisms which block access such that only users that belong to any
5 class of said view for which said computer application is produced can
6 connect or log on to said desired computer program, said control being
7 performed by requesting that a user wishing to log onto said desired
8 computer program indicate information that identifies a class of which
9 said user is an instance, and also indicate information that identifies the
10 user as an instance of said class so as to identify said user, and also
11 indicate information that is used as a password for that user so as to
12 authenticate said user.

1 43. [New-Providing A System View From The Action Selection
2 Presentation Pattern] The process of claim 40 wherein said step C creates
3 computer readable code which has the capability to control a computer to
4 provide said view of the system in such a manner that said system view
5 comprises user interaction mechanisms included in an Action Selection
6 Presentation Pattern associated with said view for which the application is
7 produced.

1 44. [New-Restricting The System View Wrt User Privileges] The process
2 of claim 43 wherein step C creates computer readable code which has the
3 capability to control a computer to restrict the user interface interaction
4 mechanisms to the ones the user who logged on is allowed to interact with
5 according to privileges established by data structures which relate the class
6 said user belongs to, playing the role of agent, with classes, playing the role of
7 servers, that determine which services of each server class will be available for
8 execution by said user and which attributes of each server class said user will
9 be able to query.

1 45. [New-Interaction With The System] The process of claim 40 wherein
2 step D comprises translating said validated Formal Language Specification
3 into computer readable code which has the capability to control a computer to
4 provide user interface interaction mechanisms which allow a user who has
5 logged on to interact with and access functionality of said computer application
6 by invoking services, executing queries and/or execution of user interface
7 interaction scenarios.

1 46. [New-Execution Of Services] The process of claim 45 wherein step D
2 comprises the steps of translating said Formal Language Specification into
3 computer readable code which has the capability to control a computer to
4 provide user interface interaction mechanisms which allow a user who has
5 logged in to interact with and access the functionality of said computer
6 application by execution of only selected services selected from a group of
7 services comprising events, local transactions and/or global transactions and
8 depending upon the identity of said user.

1 47. [New - Execution Of Events] The process of claim 46 where the steps in step
2 D of translating said Formal Language Specification to generate computer readable code
3 which controls a computer to allow a logged on user to interact with and execute the
4 functionality associated to an event comprise generating code which controls a computer
5 to provide a mechanism to construct the message associated to said event and to
6 execute said event.

1 48. [New – Execution Of Events: Construction Of Message] The process
2 of claim 47 wherein said step D includes the steps of translating said validated
3 Formal Language Specification to generate computer readable code which
4 controls a computer to provide a mechanism to construct said message
5 associated to said event comprises generating code to control a computer to

6 perform the following steps:

- 7 - providing a mechanism to identify the object on which the event
- 8 will be executed, except if said event is a creation event, in which
- 9 case this step will be omitted;
- 10 - providing mechanisms to give a value to every argument of said
- 11 event, said mechanisms further controlling that arguments that
- 12 require a value are provided a value and that any argument
- 13 whose value is provided receives a valid value.

1 49. [New- Execution Of Events: Execution Steps] The process of claim 47
2 wherein said step D includes the steps of translating said validated Formal
3 Language Specification to generate computer readable code which has the
4 capability to control a computer to provide a mechanism to execute a service
5 which is an event and comprises the following steps:

- 6 - translating said validated Formal Language Specification so as
- 7 to provide code which controls a computer to provide a
- 8 mechanism to, except in the case of a creation event, verify the
- 9 existence of said object on which said event will be executed, or
- 10 the non-existence of the object to be created in the case of a
- 11 creation event;
- 12 - translating said validated Formal Language Specification so as
- 13 to provide code which controls a computer to provide a
- 14 mechanism to, except in the case of a creation event, recover the
- 15 state of the object on which the event is executed from whatever
- 16 memory or database or repository or any other persistence
- 17 means (hereafter just "memory") to which said state of said
- 18 object has been saved;
- 19 - translating said validated Formal Language Specification so as

20 to provide code which controls a computer to provide a
21 mechanism to verify that, according to the state transition
22 diagram of the class owning said event, there is a valid state
23 transition labelled with said event being executed and for the
24 agent class to which the user belongs who logged onto said
25 desired computer program, in which case said mechanism will
26 update the state of the object on which the event is executed
27 according to said state transition diagram, and if there is no valid
28 state transition, said mechanism will produce an error message
29 causing the execution of the event to stop and roll back all
30 changes made to the state of the object on which said event is
31 executed

32 - translating said validated Formal Language Specification so as
33 to provide code which controls a computer to provide a
34 mechanism to verify that every precondition that is defined for
35 said event being executed and for said agent class to which the
36 user logged on to the computer application belongs, is satisfied,
37 and should any of said preconditions not be satisfied, then said
38 mechanism will produce an error with the error message being
39 predefined for said precondition that does not hold causing the
40 execution of the event to stop and roll back all changes made to
41 the state of said object on which said event is executed;

42 - except in the case of creation events or destruction events,
43 translating said validated Formal Language Specification so as
44 to provide code which controls a computer to provide a
45 mechanism to produce the changes of values to the variable
46 attributes of said class owning said event for which valuation
47 formulas in said functional model have been defined such that

48 said valuations relate said variable attributes with the event
 49 being executed, and wherein said mechanism applies only the
 50 changes to the variable attributes which are required by valuation
 51 formulas of valuations whose valuation condition formula
 52 evaluates to true, if any, or the change required by a valuation
 53 formula of a valuation having no valuation condition formula;
 54 - in the case of a creation event, translating said validated Formal
 55 Language Specification so as to provide code which has the
 56 capability to control a computer to provide a mechanism to
 57 assign a value to every constant or variable attribute of an object
 58 on which said creation event is executed and establishing
 59 relationships between said object on which said creation event
 60 is executed with objects of classes related with the class owning
 61 said creation event;
 62 - in the case of a destruction event, translating said validated
 63 Formal Language Specification so as to provide code which has
 64 the capability to control a computer to provide a mechanism to
 65 delete relationships of the object on which said destruction event
 66 is executed with objects said object is related to;
 67 - except in the case of destruction events, translating said
 68 validated Formal Language Specification so as to provide code
 69 which has the capability to control a computer to provide a
 70 mechanism to check that every integrity constraint defined in the
 71 class owning said event is satisfied, for the object whose state
 72 has been changed by said event, and should any of said integrity
 73 constraints not be satisfied, said mechanism will produce an
 74 error with the error message predefined for said integrity
 75 constraint that does not hold, said mechanism causing the

76 execution of the event to stop and roll back all changes made to
77 the state of the object on which the event is executed;
78 - except in the case of a destruction event, translating said
79 validated Formal Language Specification so as to provide code
80 which has the capability to control a computer to provide a
81 mechanism to save the changes made to said object on which
82 the event is executed to memory;
83 - in the case of a destruction event, translating said validated
84 Formal Language Specification so as to provide code which has
85 the capability to control a computer to provide a mechanism to
86 delete the object on which said destruction event is executed
87 from memory to which said object has been saved;
88 - except in the case of a destruction event, translating said
89 validated Formal Language Specification so as to provide code
90 which has the capability to control a computer to provide a
91 mechanism to check every trigger condition on said object on
92 which said event is executed for every trigger relationship defined
93 on said class owning said event, and wherein for every trigger
94 condition that holds, a mechanism to execute the service
95 associated to said trigger relationship by:
96 • providing a mechanism to determine the set of objects
97 on which said service associated to said trigger will be
98 executed, and
99 • executing said service on every object of said set of
100 objects by:
101 • giving a value to every argument of said service,
102 and ensuring that every argument that requires a
103 value is provided a value and that any argument

104 whose value is provided receives a valid value;
 105 and
 106 • invoking the execution of said service on said
 107 object;
 108 and
 109 - translating said validated Formal Language Specification so as
 110 to provide code which has the capability to control a computer to
 111 provide a mechanism to inform the requestor of the result of
 112 executing the event.

1 50. [NEW - EXECUTION OF LOCAL TRANSACTIONS] The process of
 2 claim 46 where the steps in step D of translating said Formal Language
 3 Specification to generate computer readable code which controls a computer
 4 to allow a logged on user to interact with and execute the functionality
 5 associated to a local transaction comprise generating code which controls a
 6 computer to provide a mechanism to construct the message associated to
 7 said local transaction and to execute said local transaction.

1 51. [New - Execution Of Local Transactions: Construction Of The
 2 Message] The process of claim 50 wherein the steps in step D of translating
 3 said Formal Language Specification to generate computer readable code
 4 which controls a computer to provide a mechanism to construct said message
 5 associated to the local transaction comprise generating code to control a
 6 computer to perform the following steps:
 7 - providing a mechanism to identify the object on which said local
 8 transaction will be primarily executed, except if said transaction
 9 is a creation service, in which case this step will be omitted;
 10 - providing mechanisms to give a value to every argument of said

11 local transaction, said mechanisms further controlling that
12 arguments that require a value are provided a value and that any
13 argument whose value is provided receives a valid value.

1 52. [New-Execution Of Local Transactions: Execution Steps] The
2 process of claim 50 wherein said step D includes the steps of translating said
3 validated Formal Language Specification to generate computer readable code
4 which has the capability to control a computer to execute a service which is a
5 local transaction and comprises the following steps:
6 - translating said validated Formal Language Specification so as to
7 provide code which controls a computer to provide a mechanism to,
8 except in the case of a creation service, verify the existence of the
9 object on which the local transaction will be executed or, in the case
10 of a creation service, verify the non-existence of the object to be
11 created;
12 - translating said validated Formal Language Specification so as to
13 provide code which controls a computer to, except in the case of a
14 creation service, provide a mechanism to recover the state of the
15 object on which the local transaction is executed from memory to
16 which the state of said object has been saved;
17 - translating said validated Formal Language Specification so as to
18 provide code which controls a computer to provide a mechanism to
19 verify that, according to the state transition diagram of the class
20 owning said local transaction, there is a valid state transition labelled
21 with said local transaction being executed and for the agent class to
22 which the user logged on to the system belongs, and, if there is such
23 a valid state transition, said mechanism will update the state of the
24 object on which the local transaction is executed according to said

25 state transition diagram, and, if there is no such valid state transition,
26 said mechanism will produce an error message causing the
27 execution of said local transaction to stop and roll back all changes
28 made to the state of said object on which the local transaction is
29 executed and of any other object the state of which has been
30 modified by the execution of said local transaction;

31 - translating said validated Formal Language Specification so as to
32 provide code which controls a computer to provide a mechanism to
33 verify that every precondition that is defined for the local transaction
34 being executed and for the agent class to which the user logged on to
35 the system belongs, is satisfied, and should any of said
36 preconditions not hold, then said mechanism will produce an error
37 with the error message defined for said precondition that does not
38 hold causing the execution of said local transaction to stop and roll
39 back all changes made to the state of the object on which said local
40 transaction is executed and of any other object the state of which has
41 been modified by the execution of said local transaction;

42 - for every service comprised in the transaction formula of said local
43 transaction being executed:

- 44 o if said service has an associated guard, translating said
45 validated Formal Language Specification so as to provide code
46 which controls a computer to provide a mechanism to check
47 that said guard associated to said service holds; and if said
48 guard does not hold, the rest of the steps associated to said
49 service will be omitted;
- 50 o translating said validated Formal Language Specification so as
51 to provide code which controls a computer to provide a
52 mechanism to determine the set of objects on which said

53 service comprised in said transaction formula will be executed
 54 and to provide a mechanism to execute said service on every
 55 object of said set of objects by:

- 56 • providing mechanisms to give a value to every argument of
 57 said service, said mechanisms further ensuring that
 58 arguments that require a value are provided a value and that
 59 any argument whose value is provided receives a valid
 60 value;
- 61 • providing a mechanism to invoke the execution of said
 62 service on said object and control the result of said
 63 execution, and should said execution result in an error,
 64 causing the execution of the local transaction to stop and roll
 65 back all changes made to the state of said object on which
 66 the local transaction is executed and of any other object the
 67 state of which has been modified by the execution of said
 68 local transaction

69 - translating said validated Formal Language Specification so as to
 70 provide code which has the capability to control a computer to provide
 71 a mechanism to check that every integrity constraint holds which is
 72 defined in the class owning the local transaction and in classes
 73 whose instances include objects the state of which has been
 74 modified by the execution of said local transaction, for any object
 75 whose state has been changed by said local transaction, and,
 76 should any of said integrity constraints not hold, said mechanism
 77 produces an error with an error message defined for said integrity
 78 constraint that does not hold and causing the execution of said local
 79 transaction to stop and roll back all changes made to the state of the
 80 object on which the local transaction is executed and of any other

81 object the state of which has been modified by the execution of said
82 local transaction;
83 - translating said validated Formal Language Specification so as to
84 provide code which has the capability to control a computer to provide
85 a mechanism to check every trigger condition on any object the state
86 of which has been modified by the execution of said local transaction,
87 for every trigger relationship defined on each class owning an object
88 the state of which has been changed by execution of said local
89 transaction, and for every trigger condition that holds, a mechanism
90 to execute the service associated to said trigger relationship by:
91 • providing a mechanism to determine the set of objects on
92 which said service associated to said trigger will be
93 executed, and
94 • providing a mechanism to execute said service on every
95 object of said set of objects by:
96 • giving a value to every argument of said service, and
97 ensuring that every argument that requires a value is
98 provided a value and that any argument whose value
99 is provided receives a valid value; and
100 • invoking the execution of said service on said object;
101 and
102 - translating said validated Formal Language Specification so as to
103 provide code which has the capability to control a computer to provide
104 a mechanism so as to inform the requestor of the result of executing
105 said local transaction.

1 53. [New-Execution Of Global Transactions] The process of claim 46
2 wherein the steps in step D of translating said Formal Language Specification

3 to generate computer readable code which controls a computer to allow a
4 logged on user to interact with and execute the functionality associated to a
5 global transaction comprise generating code which controls a computer to
6 provide a mechanism to construct the message associated to said global
7 transaction and to execute said global transaction.

1 54. [New-Execution Of Global Transactions: Construction Of Message] The
2 process of claim 53 wherein the steps in step D of translating said Formal
3 Language Specification to generate computer readable code which controls a
4 computer to provide a mechanism to construct said message associated to
5 said global transaction comprises the step of generating code which controls a
6 computer to provide mechanisms to give a value to every argument of the
7 global transaction, said mechanisms further controlling that arguments that
8 require a value are provided a value and that any argument whose value is
9 provided receives a valid value.

1 55. [New-Execution Of Global Transactions: Execution Steps] The process of
2 claim 53 wherein said step D includes the steps of translating said validated
3 formal language specification to generate computer readable code which can
4 control a computer to execute a service which is a global transaction and
5 comprises the following steps:

6 - translating said validated formal language specification so as to
7 provide code which controls a computer to provide a mechanism to
8 verify that every precondition that is defined for the global transaction
9 being executed and for the agent class the user logged on to the
10 system belongs to, is satisfied, and should any of said preconditions
11 not hold, then said mechanism will produce an error with the error
12 message being defined for said precondition that does not hold, and

- 13 causing execution of said global transaction to stop and roll back all
 14 changes made to any object the state of which has been modified by
 15 execution of said global transaction;
 16 - for every service included in a transaction formula of said global
 17 transaction being executed:
 18 o if said service has an associated guard, translating said validated
 19 formal language specification so as to provide code which controls
 20 a computer to provide a mechanism to check that said guard
 21 associated to said service holds, and if said guard does not hold,
 22 the rest of the steps associated to said service will be omitted;
 23 o translating said validated formal language specification so as to
 24 provide code which controls a computer to provide a mechanism to
 25 determine the set of objects on which said service will be executed
 26 and to provide a mechanism to execute said service on every
 27 object of said set of objects by:
 28 • providing mechanisms to give a value to every argument of
 29 said service, said mechanisms further ensuring that
 30 arguments that require a value are provided a value and that
 31 any argument whose value is provided receives a valid
 32 value;
 33 • providing a mechanism to invoke the execution of said
 34 service on said object and control the result of said
 35 execution, and should said execution result in an error,
 36 causing the execution of the global transaction to stop and
 37 roll back all changes made to the state of any object the
 38 state of which might have been modified by the execution of
 39 said global transaction;
 40 - translating said validated formal language specification so as to

4 1 provide code which can control a computer to provide a mechanism
4 2 to check that every integrity constraint holds that is defined in classes
4 3 whose instances include objects the state of which has been
4 4 modified by the execution of said global transaction, for any object
4 5 whose state has been changed by said global transaction, and
4 6 should any of said integrity constraints not hold, said mechanism
4 7 produces an error with an error message defined for said integrity
4 8 constraint that does not hold thereby causing execution of said global
4 9 transaction to stop and roll back all changes made to any object the
5 0 state of which has been modified by execution of said global
5 1 transaction;

5 2 - translating said validated formal language specification so as to
5 3 provide code which can control a computer to provide a mechanism
5 4 to check every trigger condition on any object the state of which has
5 5 been modified by execution of said global transaction, for every
5 6 trigger relationship defined on each class owning any object the state
5 7 of which has changed by execution of said global transaction, and, for
5 8 every trigger condition that holds, a mechanism to execute the service
5 9 associated to said trigger relationship by:

- 6 0 • providing a mechanism to determine a set of objects on
6 1 which said service associated to said trigger will be
6 2 executed, and
- 6 3 • providing a mechanism to execute said service on every
6 4 object of said set of objects by:
 - 6 5 • giving a value to every argument of said service, and
 - 6 6 ensuring that every argument that requires a value is
 - 6 7 provided a value and that any argument whose value
 - 6 8 is provided receives a valid value; and

69 • invoking the execution of said service on said object;
70 and
71 - translating said validated formal language specification so as to
72 provide code which can control a computer to provide a mechanism
73 so as to inform the requestor of the result of executing said global
74 transaction.

56. [New-Execution Of Queries] The process of claim 45 wherein the steps in step D of translating said formal language specification to generate computer readable code which can control a computer to allow a logged on user to interact with and execute the functionality of a query as part of filter patterns independently or in the context of a Class Population Presentation Pattern comprise generating computer code to control said computer to provide a mechanism to construct the message associated with said filter and to execute the query associated with said filter.

57. [NEW-EXECUTION OF QUERIES: CONSTRUCTION OF MESSAGE] The process of claim 56 wherein said step D includes the steps of translating said validated formal language specification to generate computer readable code which can control a computer to provide a mechanism to construct said message associated to the filter comprise generating code to control a computer to perform the following steps:

- providing a mechanism to indicate a filter whose associated query is to be executed;
- providing mechanisms to give a value to every filter variable of a filter pattern of said filter, said mechanisms further controlling that filter variables that require a value are provided a value and that any filter variable whose value is provided receives a valid value;

- 13 - providing a mechanism to indicate a display set pattern to be used in
- 14 performing the query
- 15 - providing a mechanism to indicate an order criterion pattern, if any, to
- 16 be used in performing said query.

1 58. [New-Execution Of Queries: Execution Steps] The process of claim 57

2 wherein said step D includes the steps of translating said validated formal

3 language specification to generate computer readable code which can control

4 a computer to provide a mechanism to execute the query associated with said

5 filter and comprises the following steps:

- 6 - translating said validated formal language specification so as to
- 7 provide code which controls a computer to provide a mechanism to
- 8 access the population of instances of the class owning said filter
- 9 pattern from memory;
- 10 - translating said validated formal language specification so as to
- 11 provide code which controls a computer to provide a mechanism to
- 12 retrieve instances of said population which fulfil the condition stated
- 13 by the filter formula of said filter depending on the values assigned to
- 14 every filter variable of said filter;
- 15 - translating said validated formal language specification so as to
- 16 provide code which controls a computer to provide a mechanism to
- 17 access only part of the state of every instance of said population
- 18 matching said condition stated by said filter formula of said filter, said
- 19 part of said state being defined said display set selected in said
- 20 message, said part of said state dictated by said display set being
- 21 further constrained to the attributes said user logged onto said
- 22 desired computer program is allowed to query;
- 23 - translating said validated formal language specification so as to

2 4 provide code which controls a computer to provide a mechanism to
2 5 return the instances of said population which fulfil said condition
2 6 stated by said filter formula and, if an order criterion pattern has been
2 7 indicated in said message, return said instances of said population
2 8 which fulfil said condition stated by said filter formula in the order
2 9 stated by said order criterion pattern.

1

1 59. [New-Execution Of Interaction Scenarios] The process of claim 45
2 wherein step D comprises the steps of translating said formal language
3 specification into computer readable code which can control a computer to
4 provide user interface interaction mechanisms which allow a user who has
5 logged on to execute services and queries by interacting with one or more user
6 interface interaction scenarios implemented as Service Presentation Patterns,
7 Instance Presentation Patterns, Class Population Presentation Patterns and/or
8 Master/Detail Presentation Patterns.

1

2 60. [New-Execution Of Service Presentation Patterns] The process of claim
3 59 wherein the steps in step D of translating said validated formal language
4 specification to generate computer readable code which can control a
5 computer to allow a logged on user to execute a service by interaction with a
6 Service Presentation Pattern related to said service, said steps of translating
7 comprising steps to generate computer readable code which implements a
8 set of mechanisms that collaborate to construct a message for the execution of
9 the service associated to said Service Presentation Pattern, including:

9 - translating said validated formal language specification so as to
10 provide code which controls a computer to provide a mechanism to
11 identify said Service Presentation Pattern by its alias;

- 12 - translating said validated formal language specification so as to
13 provide code which controls a computer to provide a mechanism to
14 present a help message associated with said Service Presentation
15 Pattern to said user interacting with said Service Presentation
16 Pattern;
- 17 - translating said validated formal language specification so as to
18 provide code which controls a computer to provide a mechanism to
19 identify each argument of said service associated with said Service
20 Presentation Pattern by the alias of each said argument of said
21 service;
- 22 - translating said validated formal language specification so as to
23 provide code which controls a computer to provide a mechanism to
24 present each argument of said service associated with said Service
25 Presentation Pattern to said user interacting with said Service
26 Presentation Pattern in the order and groups dictated by the
27 Arguments Grouping Presentation Pattern (if any) associated to said
28 Service Presentation Pattern;
- 29 - translating said validated formal language specification so as to
30 provide code which controls a computer to provide a mechanism to
31 let the user provide a value for each argument of the service
32 associated to said Service Presentation Pattern
- 33 - translating said validated formal language specification so as to
34 provide code which controls a computer to provide a mechanism to
35 ensure that every argument of said service associated with said
36 Service Presentation Pattern that requires a value has a value and to
37 validate that every argument that has a value has a valid value
38 according to every said argument data type and said Introduction
39 Pattern, if any, associated with every said argument;

- 4 0 - translating said validated formal language specification so as to
- 4 1 provide code which controls a computer to provide a mechanism to
- 4 2 present the user with the default value, if any, of every argument of
- 4 3 said service associated to said Service Presentation Pattern;
- 4 4 - translating said validated formal language specification so as to
- 4 5 provide code which controls a computer to provide a mechanism to
- 4 6 access an objects selection mechanism corresponding to a
- 4 7 Population Selection Pattern, if any, associated with every object
- 4 8 valuated argument of the service associated with said Service
- 4 9 Presentation Pattern;
- 5 0 - translating said validated formal language specification so as to
- 5 1 provide code which controls a computer to provide a mechanism to
- 5 2 present said user with a help message, if any, associated to every
- 5 3 argument of said service associated to said Service Presentation
- 5 4 Pattern;
- 5 5 - translating said validated formal language specification so as to
- 5 6 provide code which controls a computer to provide a mechanism to
- 5 7 implement a Dependency Pattern, if any, associated with every
- 5 8 argument of the service associated with said Service Presentation
- 5 9 Pattern, said mechanism:
- 6 0 o controlling a computer to monitor the occurrence of said user
- 6 1 interface interaction events relevant to every said argument, which
- 6 2 either change the value of said argument or activate/deactivate said
- 6 3 argument;
- 6 4 o controlling a computer so as to check that the condition of an
- 6 5 Event-Condition-Action (hereafter ECA) rule of said Dependency
- 6 6 Pattern holds, and
- 6 7 o executing the actions in said ECA rule of said Dependency Pattern

- 68 to assign a value and/or activate and/or deactivate other arguments
69 of the service associated to said Service Presentation Pattern;
70 - translating said validated formal language specification so as to
71 provide code which controls a computer to provide a mechanism to
72 present said user the elements in a Display Set Pattern, if any,
73 assigned as a Supplementary Information Pattern to every object
74 valuated argument of said service associated with said Service
75 Presentation Pattern, whenever the value of every said argument
76 changes;
77 - translating said validated formal language specification so as to
78 provide code which controls a computer to provide a mechanism to
79 allow a user to cancel the interaction of said user with said
80 interaction scenario represented by said Service Presentation
81 Pattern;
82 - a mechanism to confirm and send the message which will cause the
83 service associated to said Service Presentation Pattern to execute.

- 1 61. [New-Execution Of Instance Presentation Patterns] The process of
2 claim 92 wherein the steps in step D of translating said validated formal
3 language specification to generate computer readable code which can control
4 a computer to allow a logged on user query information on an instance of a
5 class, execute services on said instance and/or navigate to interaction
6 scenarios displaying information related with said instance, by interaction with
7 an Instance Presentation Pattern of the class owning said instance, said steps
8 of translating comprising the following steps:
9 - translating said validated formal language specification so as to
10 provide code which controls a computer to provide a mechanism to
11 identify said Instance Presentation Pattern by its alias;

- 12 - translating said validated formal language specification so as to
- 13 provide code which controls a computer to provide a mechanism to
- 14 present a help message, if any, associated with said Instance
- 15 Presentation Pattern;
- 16 - translating said validated formal language specification so as to
- 17 provide code which controls a computer to provide a mechanism to
- 18 present said user with the value of each element in the Display Set
- 19 Pattern of said Instance Presentation Pattern, each element identified
- 20 by the alias of said element and presented in the order dictated by
- 21 said Display Set Pattern, said elements being further restricted to
- 22 those elements that correspond to attributes the logged on user is
- 23 allowed to query;
- 24 - translating said validated formal language specification so as to
- 25 provide code which controls a computer to provide a mechanism to
- 26 access each user interface interaction scenarios corresponding to
- 27 each service that can be executed on the object displayed by said
- 28 Instance Presentation Pattern, each of said services identified by the
- 29 alias of the Service Presentation Pattern corresponding to each of
- 30 said services, said services being further restricted to the ones the
- 31 logged on user is allowed to execute;
- 32 - translating said validated formal language specification so as to
- 33 provide code which controls a computer to provide a mechanism to
- 34 present said user with a help message associated with each
- 35 service, if any, executable on said object displayed by said Instance
- 36 Presentation Pattern;
- 37 - translating said validated formal language specification so as to
- 38 provide code which controls a computer to provide a mechanism to
- 39 access each user interface interaction scenario corresponding to

each class, if any, owning instances related with the instance belonging to the class owning said Instance Presentation Pattern, each of said user interface interaction scenarios being identified by the alias of its corresponding Presentation Pattern, said user interface interaction scenarios being further restricted to the ones corresponding to classes said logged on user is allowed to query;

- translating said validated formal language specification so as to provide code which controls a computer to provide a mechanism to present said user with any help message associated with each of the user interface interaction scenarios corresponding to each class, if any, owning instances related with instance belonging to the class owning said Instance Presentation Pattern; and

- translating said validated formal language specification so as to provide code which controls a computer to provide a mechanism to cancel the interaction of the logged on user with the user interface interaction scenario represented by said Instance Presentation Pattern.

62. [New-Execution Of Class Population Presentation Patterns] The process of claim 59 wherein the steps in step D of translating said validated formal language specification to generate computer readable code which can control a computer to allow a logged on user to query information on a set of instances of a class, execute services on any instance in said set and or navigate to interaction scenarios displaying information related with any said instance of said set, by interaction with a Class Population Presentation Pattern of the class owning said set of instances, said steps of translating comprising the following steps:

- translating said validated formal language specification so as to

11 provide code which controls a computer to provide a mechanism to
12 identify said Class Population Presentation Pattern by its alias;
13 - translating said validated formal language specification so as to
14 provide code which controls a computer to provide a mechanism to
15 present any help message associated with said Class Population
16 Presentation Pattern;
17 - translating said validated formal language specification so as to
18 provide code which controls a computer to provide a set of
19 mechanisms to obtain and display a set of instances of the class
20 owning said Class Population Presentation Pattern, comprising:
21 o a mechanism to identify the Filter Pattern, if any, associated to
22 said Class Population Presentation Pattern;
23 o a mechanism to identify and select one of the Order Criterion
24 Patterns, if any, associated to said Class Population
25 Presentation Pattern;
26 o a mechanism to identify each Filter Variable, if any, of said Filter
27 Pattern, if any, associated to said Class Population Presentation
28 Pattern, said identification by the alias of each of said Filter
29 Variables, and to present said Filter Variables to said user in the
30 order they are defined in said Filter Pattern;
31 o a mechanism to present said user a default value, if any, of each
32 Filter Variable, if any, of said Filter Pattern, if any, associated to
33 said Class Population Presentation Pattern;
34 o a mechanism to let said user provide a value for each Filter
35 Variable, if any, of said Filter Pattern, if any, associated to said
36 Class Population Presentation Pattern;
37 o a mechanism to validate the value assigned to each Filter
38 Variable, if any, of said Filter Pattern, if any, associated to said

39 Class Population Presentation Pattern, said validation carried
40 out according to:

- 41 • the data type of said Filter Variable;
- 42 • what is dictated by the Introduction Pattern, if any,
- 43 associated to said Filter Variable;
- 44 o a mechanism to access the objects selection mechanism
45 corresponding to a Population Selection Pattern which defines
46 the process of observing and selecting objects in a multiple
47 objects society, said Population Selection Pattern being
48 associated to every object valuated Filter Variable, if any, of said
49 Filter Pattern, if any, associated to said Class Population
50 Presentation Pattern;
- 51 o a mechanism to present said user with a help message, if any,
52 associated to each Filter Variable, if any, of said Filter Pattern, if
53 any, associated to said Class Population Presentation Pattern;
- 54 o a mechanism to present said user with elements in any Display
55 Set Pattern assigned as a Supplementary Information Pattern to
56 every object valuated Filter Variable, if any, of said Filter Pattern, if
57 any, associated to said Class Population Presentation Pattern,
58 whenever the value of every said Filter Variable changes;
- 59 o a mechanism to invoke execution of the query represented by
60 said Filter Pattern, if any, associated with said Class Population
61 Presentation Pattern, or to invoke the retrieval of the full
62 population of said class owning said Class Population
63 Presentation Pattern;
- 64 - translating said validated formal language specification so as to
65 provide code which controls a computer to provide a mechanism to
66 display the value of each element, for every instance in the

67 population of the class or for every instance returned as a result of
 68 executing said query represented by said Filter Pattern, if any,
 69 associated with said Class Population Presentation Pattern, said
 70 each element being in the Display Set Pattern associated to said
 71 Class Population Presentation Pattern, said each element the
 72 value of which is displayed being restricted to those elements
 73 corresponding to attributes the logged on user is allowed to query;
 74 - translating said validated formal language specification so as to
 75 provide code which controls a computer to provide a mechanism to
 76 select one of the objects or instances presented to the user by
 77 said Class Population Presentation Pattern;
 78 - translating said validated formal language specification so as to
 79 provide code which controls a computer to provide a mechanism
 80 to, upon selection of one of said objects presented by said Class
 81 Population Presentation Pattern, access each of said user
 82 interface interaction scenarios corresponding to each service that
 83 can be executed on said selected object, each of said services
 84 identified by the alias of the Service Presentation Pattern
 85 corresponding to each of said services, said services further
 86 restricted to the services the logged on user is allowed to execute;
 87 - translating said validated formal language specification so as to
 88 provide code which controls a computer to provide a mechanism to
 89 present said user with any help message associated with each
 90 service executable on said selected object;
 91 - translating said validated formal language specification so as to
 92 provide code which controls a computer to provide a mechanism
 93 to, upon selection of one of said objects presented by said Class
 94 Population Presentation Pattern, access each of said user

95 interface interaction scenarios corresponding to each class, if any,
 96 owning instances related with said selected object of said class
 97 owning said Class Population Presentation Pattern, each of said
 98 user interface interaction scenarios being identified by the alias of
 99 its corresponding Presentation Pattern, said user interface
 100 interaction scenarios being further restricted to the ones
 101 corresponding to classes said logged on user is allowed to query;
 102 - translating said validated formal language specification so as to
 103 provide code which controls a computer to provide a mechanism to
 104 present said user any help message associated with each of said
 105 user interface interaction scenarios corresponding to each class, if
 106 any, owning instances related with said selected object of said
 107 class owning said Class Population Presentation Pattern; and
 108 - translating said validated formal language specification so as to
 109 provide code which controls a computer to provide a mechanism to
 110 cancel the interaction of the logged on user with the interface
 111 interaction scenario represented by said Class Population
 112 Presentation Pattern.

1 63. [NEW-EXECUTION OF MASTER/DETAIL PRESENTATION PATTERNS] The
 2 process of claim 59 wherein the steps in step D of translating said validated
 3 formal language specification to generate computer readable code which can
 4 control a computer to allow a logged on user to interact with an interaction
 5 scenario corresponding to a Master/Detail Presentation pattern so as to query
 6 information on:
 7 - instances belonging to the class owning said Master/Detail
 8 Presentation Pattern as presented in a Presentation Pattern
 9 referred to as a Master Presentation Pattern, which is either an

10 Instance Presentation Pattern or a Class Population Presentation
11 Pattern owned by the same class owning said Master/Detail
12 Presentation Pattern, and
13 - instances belonging to other classes related to said class owning
14 said Master/Detail Presentation Pattern, each set of related
15 instances as presented by a Presentation Pattern referred to as
16 Detail Presentation Pattern, which is either an Instance
17 Presentation Pattern, or a Class Population Presentation Pattern,
18 or belonging to a Master/Detail Presentation Pattern owned by said
19 class related with said class owning said Master Presentation
20 Pattern,
21 said steps of translating comprising the steps of:
22 - translating said validated formal language specification so as to
23 provide code which controls a computer to provide a mechanism to
24 identify said Master/Detail Population Presentation Pattern by its
25 alias;
26 - translating said validated formal language specification so as to
27 provide code which controls a computer to provide a mechanism to
28 present said user any help message associated with said
29 Master/Detail Presentation Pattern;
30 - translating said validated formal language specification so as to
31 provide code which controls a computer to provide a mechanism to
32 present said user the Master Presentation Pattern of said
33 Master/Detail Presentation Pattern;
34 - translating said validated formal language specification so as to
35 provide code which controls a computer to provide a mechanism to
36 present said user with every Detail Presentation Pattern of said
37 Master/Detail Presentation Pattern;

- 38 - translating said validated formal language specification so as to
39 provide code which controls a computer to provide a mechanism to
40 synchronize information displayed in every Detail Presentation
41 Pattern whenever:
42 o the object displayed in the Master Presentation Pattern changes,
43 or
44 o the selection of one of the objects displayed in the Master
45 Presentation Pattern changes
46 - translating said validated formal language specification so as to
47 provide code which controls a computer to provide a mechanism to
48 cancel the interaction of the logged on user with the interaction
49 scenario represented by said Master/Detail Presentation Pattern.
50
51